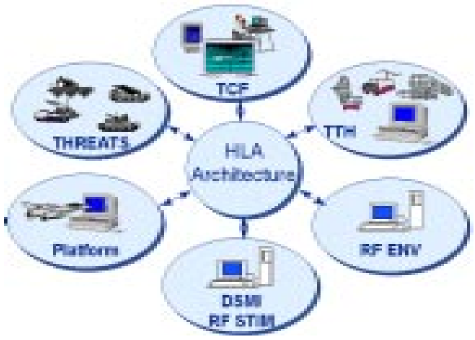
Tuning RTI Performance for an EW T&E Federation

*A Technical Paper from the Joint **A**dvanced **D**istributed **S**imulation Joint Test Force*



Mr. Clyde Harris
Mr. Jerry Black
Science Applications International Corporation

Presented at the International Test and Evaluation Association
Modeling and Simulation Workshop
7-10 December 1998, Las Cruces, NM

JADS JTF
http://www.jads.abq.com
11104 Menaul NE
Albuquerque, NM 87112-2454
(505) 846-1291
FAX (505) 846-0603

Tuning RTI Performance for an EW T&E Federation

Mr. Clyde Harris
Mr. Jerry Black
Science Applications International Corporation
JADS JTF
11104 Menaul Blvd. NE
Albuquerque, NM 87112
505-846-0909, 505-846-0467
black@jads.kirtland.af.mil, harris@jads.kirtland.af.mil

Keywords:
RTI testing, Latency, T&E, Electronic Warfare, Performance-oriented Federation, RTI test software

ABSTRACT: *The Joint Advanced Distributed Simulation (JADS) Joint Test Force (JTF) is chartered by the Office of the Secretary of Defense to investigate the utility of Advanced Distributed Simulation (ADS) Technology, including the High Level Architecture (HLA), to Test and Evaluation (T&E). JADS is executing three formal test programs, including the Electronic Warfare (EW) test, representing slices of the overall T&E spectrum to form its conclusions. The EW test is using HLA and the runtime infrastructure (RTI) version 1.3 to support two electronic warfare test events using a distributed cross-country network linking an instrumented EW system, constructive models, and virtual simulations. The JADS federation links six federates on six host computers passing attributes and interactions within constrained timing tolerances representing opposing EW systems operating in a live test environment.*

Prior to executing formal EW test events, JADS set up a test bed of federate host computers and long haul communications components to be used for the EW test events. Systematic test procedures were used to gather a series of performance benchmarks for the computer hosts, network components, and RTI versions 1.02 and 1.3. Testing started with basic network tests using two directly connected computers followed by adding communications components to the tests. The RTI and a test federate were then installed on each computer. Tests progressed with incremental addition of computers and test federates up to a total of six federates on six computers. When the JADS EW federate models and long-haul network became available, final tests and benchmarks were accomplished. This test methodology provided a series of filter screens designed to quantify RTI performance while isolating increasingly complex federate interactions and implementation issues for detailed examination.

These software tools and test procedures were designed to verify the latency characteristics of the JADS architecture prior to and after installation of the long haul T-1 network, implementing the actual EW test federates, and performing the formal EW test events. This paper describes the results of measuring latency, steps in tuning the network and RTI, and lessons learned from using a set of test cases oriented to JADS federation performance requirements and RTI 1.3 software.

JADS Federation Description

The JADS EW test uses T-1 circuits, and communications and encryption devices to link three test locations in different states. The test links constructive and virtual simulations to reproduce an EW test environment on an open air range (OAR). In addition to JADS, two key EW test facilities are involved: the Air Force Electronic Warfare Environment Simulator (AFEWES) and the Air Combat Environment Test and Evaluation Facility (ACETEF). Three network nodes interconnect a total of six federates representing critical components of the OAR test environment including the test aircraft, aircraft EW systems, and threat systems. Four of the six federates execute on dedicated SGI O2 workstations in the JADS test control facility at Albuquerque, New Mexico. There is one federate executing on an SGI O2 at the ACETEF in Patuxent River, Maryland, and one federate executing on an SGI Challenge at the AFEWES hardware-in-the-loop facility in Fort Worth, Texas. The federates at JADS in Albuquerque will publish a combined 2 attributes at 20 Hz (i.e., 20 messages/second). The worst case instance of the AFEWES federate will have 11 attributes published at 20 Hz. The ACETEF federate will publish 1 attribute at 20 Hz. All nodes will publish interactions at approximately 1 Hz. The largest JADS federation attribute or interaction is 106 bytes not including the overhead bytes added by network protocols or the RTI. One execution of the JADS federation replicating a pass on the OAR will take about four minutes. The test federation and network architecture being used by JADS is illustrated in figure 1. The JADS test bed uses the same computer and communications components that will be installed for the formal EW test events.

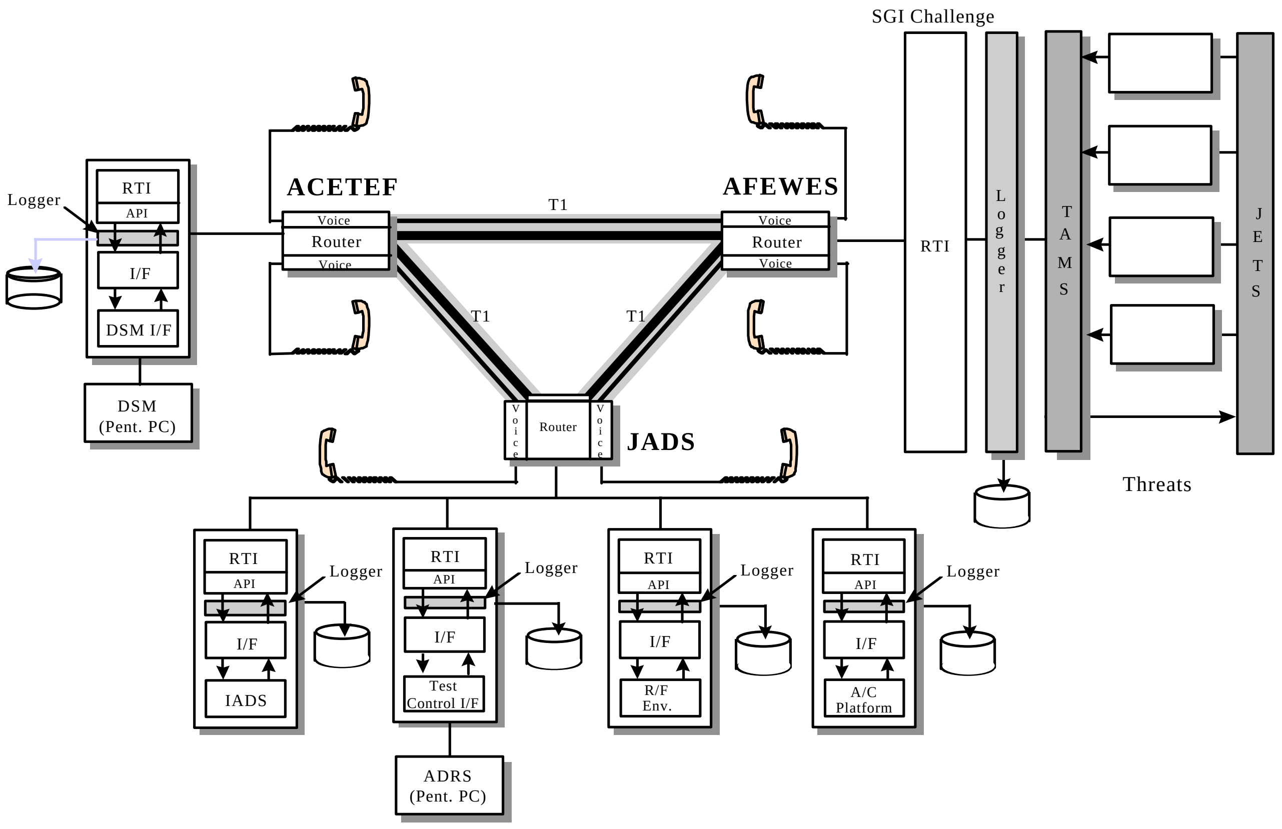


Figure 1. JADS EW Test Architecture and Federates

Due to the stringent performance requirements for testing EW systems, JADS identified a maximum "end-to-end" latency budget of 500 milliseconds(ms) round-trip including facilities, networks, and software, and the RTI[1]. This budget does not include actions taken by the federate, e.g., the digital system model (DSM) or AFEWES. Of this 500 ms, the RTI is allocated 140 ms round-trip (70 ms each way). JADS selected the HLA to use for the EW test to include it in the JADS evaluation of the utility of ADS for T&E. Early on, JADS and DMSO began working together on the use of HLA for the EW tests. To maximize success in using the RTI, JADS began work with DMSO to articulate our EW test design and the need for a "performance-oriented" RTI in May 1997. The primary tool for documenting and communicating JADS requirements to DMSO and the RTI development community evolved into the Federation Execution Planners Workbook [2].

Test Software

There are two types of software JADS developed for the RTI tests. First, we developed software to send data one way between two computers. There are versions of this software that perform tests without an RTI (both TCP and IP multicast), and there are versions that perform tests using the RTI (reliable and best effort). RTI reliable traffic is sent using TCP transmission. RTI best effort traffic is sent using IP Multicast transmission. Using the JADS test bed configuration, the non-RTI tests give an approximation of the raw network performance. The purpose of this software is to characterize the network in the simplest of cases. The second type of software we developed was an RTI federate capable of running in different configurations on multiple computers within a federation execution. The purpose of this software is to determine how the RTI performs in a more realistic environment under loads anticipated for the JADS federations.

In all JADS tests, latency and lost data are the two metrics examined. To track lost data, all of our messages (either attributes or interactions) contain a serial number. To calculate latency, the send time is included in the message. When a message arrives, the receive time is saved with the send time to be used to calculate the latency. For this design to work, the system time for all the computers that participate in a test must be synchronized. In the JADS test federation, we will be using Datum BanComm GPS cards to accept an IRIG B or GPS input to synch the time. These cards were not available when we began RTI testing so we used the Network Time Protocol (xntp) software to synchronize the clocks on all test computers.

We have a GPS receiver that provides time to one of the computers via its serial port. This computer is the Stratum-1 time server. All of the other computers in the network receive their time from the time server. It takes a few days to get the whole system initially configured and settled down. After that, the system time on all computers remains within 1 ms of GPS time. The xntp software generates statistics on how well it is keeping time. We used a Datum BanComm card to verify that the offset reported by the xntp software was accurate.

Two Node Test Description

The RTI test hardware configurations progressively increase in complexity until the entire federation and network architecture (except for the T1 lines) is in place in the JADS test bed. Starting with a two computer point-to-point configuration, we gathered basic performance data for network IP Multicast data, network TCP data, RTI 1.0-2 best effort data, RTI 1.0-2 reliable data, RTI 1.3 beta (1.3b) best effort data, RTI 1.3b reliable data, RTI 1.3-2 Early Access Version (RTI 1.3-2EAV) reliable data, and RTI 1.3-2 (early official release) reliable data.

The two node test configuration is shown in figure 2. The test configuration included all network components using a two-node network for the same series of tests. The associated communications link and hardware/software configuration are also being tested. All sources of possible latency were computed through a disciplined process of adjusting one variable at a time and collecting recorded time data for the same message type in differing reference test conditions. The two-node network test used an SGI O2 5000 and an SGI O2 10000 running IRIX 6.3. The test software and RTI were hosted on each computer for all tests using this configuration.

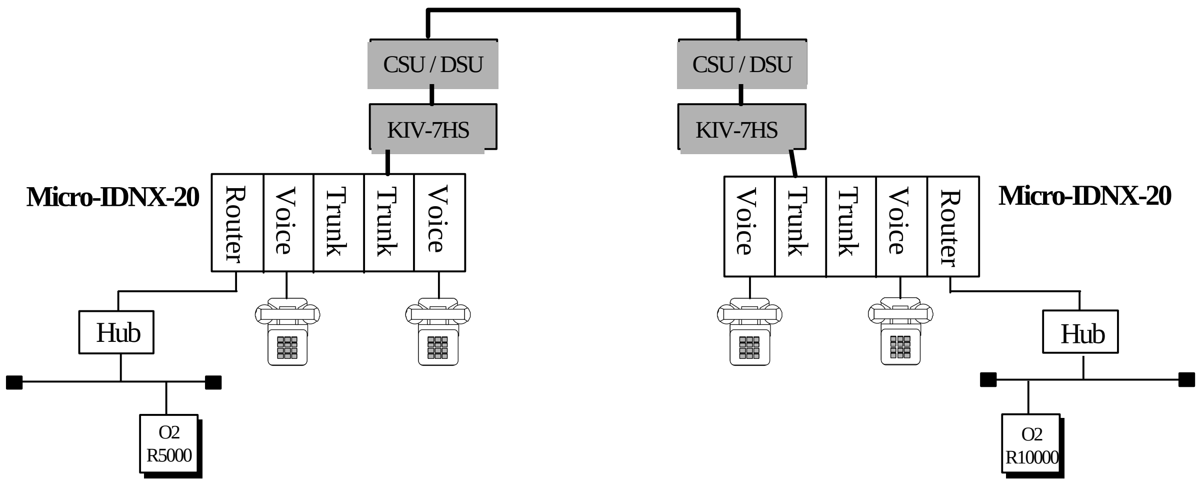


Figure 2. JADS 2 node RTI test bed configuration with communications devices

Standard Test Methodology for Two-Node Test

(1) Baseline the hardware configuration performance without RTI.
(2) Install RTI software.
(3) Run attribute size tests, attribute rate tests, interaction size, RTI polling interval (and duration) tests using best effort transport with multicasting.
(4) Add network communications hardware components.
(5) Repeat steps 1 through 4 for the second configuration.

(6) Compare latency data for different hardware/RTI software configurations. Attribute and interaction message rates, sizes, and tick were each examined around the values specified in the JADS Federation Execution Planners Workbook.

One -Way Software for Two-Node Tests

The one-way software is designed to exercise the network and the RTI with different data sizes and transmit rates. The size is varied between 17, 51, 101, 301, 501, and 1001 bytes. The transmit rate is varied between 5, 10, 20, 50, 100, 200, 400, and 500 Hz. The complete matrix of rate and size combinations was tested. Each test case, defined by a specific rate and size pair, ran for thirty seconds. For the RTI version of the one-way software, a separate matrix was generated for attributes sent as reliable and best effort. Initially, only one test of each pair was run.

There are two programs that must be run in the one-way, network only (no RTI) tests – a sender and a receiver. The programs used for the JADS tests are: *tcp_sender, tcp_receiver*, *ipmc_sender*, and *ipmc_receiver*. To generate a test matrix, first start the receiver on one computer (the “to” computer). Then, start the sender on another computer (The *tcp_sender* program requires that you specify the node name of the computer upon which the receiver is running). The sender then loops through each test case of size and rate, sending data to the receiver. At the start of each test case, the sender transmits a **start** message to the receiver indicating the size, rate and total count of messages to be sent. This information is used by the receiver to name the output file and to determine if any messages were lost. After sending the control message, the sender transmits the data. The data message contains a sequential serial number and the time (i.e., when it was time-tagged in the sending code) the message was sent. When a message arrives at the receiver the system time is obtained. The receiver stores the time sent and time received in an array indexed by the serial number. After sending all of the data for a test case, the sender transmits an **end** message.

When the receiver gets the **end** message, all of the data from the test case is written to the data file. To eliminate its effect on the latency calculation, no I/O occurs while the data is being transmitted. The data file contains a record for each message that should have been received. If the message was received, the serial number, send time, receive time, and latency are written to the file. Prior to each test case, the receiver initializes the start times to zero. At the end of a test case, if the send time is zero for a serial number, then that message was not received. In this case, the serial number and the word MISSING are written to the output file. A summary file is also created by the receiver. There is a record in the summary file for each test case that was run. The record contains the data filename followed by the minimum, maximum, and mean latency for the test case.

This sequence of steps is repeated for every combination of size and rate. Due to the fact that the some of the high data rate and size combinations disrupt the network, the sender waits 5 seconds between test cases. When all test cases have been run, an additional **end** message is transmitted by the sender indicating that the test is done.

There is only one federate program used for the one-way RTI tests. It is called *test*. *Test* accepts command line parameters that tell it to run as the master (-m) or the slave (-s). To generate an RTI test matrix, first start the *test* federate as a slave on one computer. After a message is displayed that the slave is waiting for data, start the *test* federate as a master on another computer. The processing steps for the *test* federate are the same as the steps for the network tests. It produces data files and a summary file in the same format as the network software.

One-way Test Results

For the network IP Multicast tests (no RTI), the minimum, maximum, and mean latencies were between 7 and 20 ms until we began sending 301 bytes at 500 Hz (a much higher rate than we will be transmitting in the JADS federation). There were no lost messages in the tests below 301 bytes at 500 Hz. In the tests that lost data, the latencies became over 100 ms (up to 450 ms).

For the network TCP tests (no RTI), there is a significant increase in the latency once you transmit at rates greater than 5 Hz. There are also large variations between the minimum and maximum latencies among the test cases. At transmit rates of 5 Hz, the minimum latencies were between 7 and 15 ms, the maximum latencies were between 20 and 35 ms, and the mean latencies were between 8 and 17 ms. At transmit rates above 5 Hz, the minimum latencies were 8 - 15 ms, the maximums were 200 - 500 ms, and the mean latencies were between 50 and 400 ms. It becomes clear from looking at a plot of the data from one trial (Figure 3 below) that the data is being buffered somewhere. Upon further investigation, we determined that the buffering was caused by implementation of the Nagle algorithm [3]. The Nagle algorithm buffers small packets on the transmit side until an ACK is received from the previous transmit. On SGI computers, the network can wait up to 200 ms before sending the buffered packets. This explains the jump in latency at transmit rates over 5 Hz. By default, TCP sockets on SGIs run with the Nagle algorithm enabled. To disable the Nagle algorithm, specify TRUE for the socket option TCP_NODELAY.

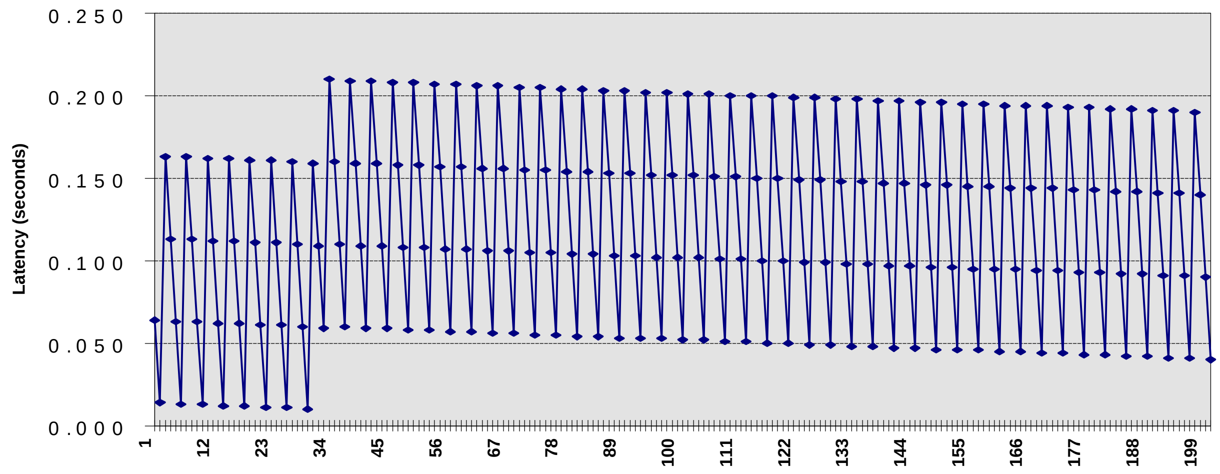


Figure 3. TCP Latency 101 bytes at 20 Hz

With the Nagle algorithm disabled, minimum and mean latencies were all between 7 and 20 ms up until the sender started transmitting 301 bytes at 500 Hz. In these tests, some of the maximums were as high as 100 ms. The cause of these intermittent spikes is still being investigated. For the tests transmitting 301 bytes (or more) at 500 Hz, the maximum latencies were more than 3 seconds with mean values of around 500 ms.

In the RTI 1.0-2 best effort tests, the latencies were slightly higher than the network IP Multicast tests (mean latencies between 9 and 24 ms). Just as in the Multicast tests, the receiver began to lose data and latencies increased when the sender transmitted 301 bytes at 400 Hz.

The RTI 1.0-2 reliable tests also show the effects of the Nagle algorithm, but the latencies are much higher than those of the TCP network tests. At transmit rates above 5 Hz, the maximum latencies were between 170 ms and 1 second, and the mean latencies were between 70 and 300 ms.

In the RTI 1.3b best effort tests, data loss occurred with smaller packet sizes than in the 1.0-2 tests. This was due to the fact that RTI 1.3b data packet headers were 400 bytes long. Data packet headers in subsequent versions of the RTI are approximately 80 bytes long. Other than that, the results were similar to the 1.0-2 tests.

The RTI 1.3b reliable tests still show the effects of the Nagle algorithm. It wasn't until after we ran the RTI 1.3b tests that we discovered the problem with the Nagle algorithm and how to disable it. With this version of the RTI, the reliable traffic was having problems even at 5 Hz. The maximum latency for the 5 Hz tests was between 200 and 300 ms, and the mean latency was between 40 and 120 ms. Also, for the tests using larger sizes at higher data rates (e.g. 301 bytes at 500 Hz), the maximum latencies were between 10 and 55 seconds, and the mean latencies were between 4 and 55 seconds.

We provided our RTI 1.3b results to DMSO along with the information we learned regarding the Nagle algorithm and the TCP_NODELAY socket option. The RTI 1.3 development team modified the RTI to disable the Nagle algorithm for all reliable traffic. They also incorporated other modifications into RTI 1.3-2EAV with the objective of improving performance for reliable traffic.

In the RTI 1.3-2EAV reliable tests (with the Nagle algorithm now disabled), the performance of reliable traffic drastically improved. With the sender publishing up to 301 bytes at 400 Hz, the minimum latencies were between 8 and 12 ms, and the maximum latencies were between 10 and 200 ms. Once again, we observed intermittent latency spikes that cause problems. When the master federate tried to publish 301 bytes at 400 Hz, reliable data was lost. When it tried to publish 501 bytes at 400 Hz, the slave federate crashed. These problems never occurred in previous tests of RTI versions. However, they are outside of the JADS expected performance requirement. Figure 4 shows the RTI 1.3-2 reliable test matrix.

The results are similar to the 1.3-2EAV results. The RTI had problems when the federate began transmitting 301 bytes at 400 Hz. It eventually terminated with an RTI internal exception.

RTI 1.3-2 Reliable

Minimum Latency (sec)						
	Packet Size (bytes)					
Rate(Hz)	**17**	**51**	**101**	**301**	**501**	**1001**
5	0.008	0.008	0.008	0.010		
10	0.008	0.008	0.008	0.010		
20*	0.007	0.008	0.008	0.010		
50	0.007	0.008	0.008	0.010		
100	0.007	0.008	0.008	0.010		
200	0.007	0.008	0.008	0.010		
400	0.007	0.008	0.008	**		
500	0.007	0.008	0.008			

Maximum Latency (sec)						
	Packet Size (bytes)					
Rate(Hz)	**17**	**51**	**101**	**301**	**501**	**1001**
5	0.009	0.009	0.009	0.011		
10	0.031	0.009	0.014	0.011		
20*	0.009	0.010	0.013	0.039		
50	0.011	0.010	0.010	0.013		
100	0.011	0.017	0.015	0.023		
200	0.086	0.014	0.019	0.012		
400	0.037	0.019	0.073	**		
500	0.023	0.057	0.170			

Mean Latency (sec)						
	Packet Size (bytes)					
Rate(Hz)	**17**	**51**	**101**	**301**	**501**	**1001**
5	0.008	0.008	0.009	0.010		
10	0.008	0.008	0.009	0.010		
20*	0.008	0.008	0.008	0.010		
50	0.008	0.008	0.008	0.010		
100	0.008	0.008	0.008	0.010		
200	0.008	0.008	0.008	0.010		
400	0.008	0.008	0.009	**		
500	0.008	0.008	0.010			

Notes: * Values within the border indicate JADS EW expected rates and sizes
** The slave had problems receiving 301-bytes @ 400 Hz and above

Figure 4. RTI 1.3-2 Reliable Test Matrix

Three Node Test Description

These tests were designed to assist JADS in optimizing the performance of the RTI as well as the JADS EW phase 2 test federation components. The major objective of these tests was to establish the performance baseline for the RTI and provide necessary feedback to JADS management as well as the RTI developers. Once the RTI version 1.3 performance baseline is determined by JADS testers, further testing, integration, and tuning of all federation components will be performed supporting phase 2 implementation. These tests are the final benchmarks prior to the implementation and testing of actual phase 2 test software federates with the AFEWES surrogate federate during August 1998.

The test environment expanded from the two node configuration and used at least 3 and as many as 6 SGI O2 workstations (either R5000 or R10000 models) running IRIX 6.3. The three node test configuration in the EW test bed with three SGI computers is shown in figure 5.

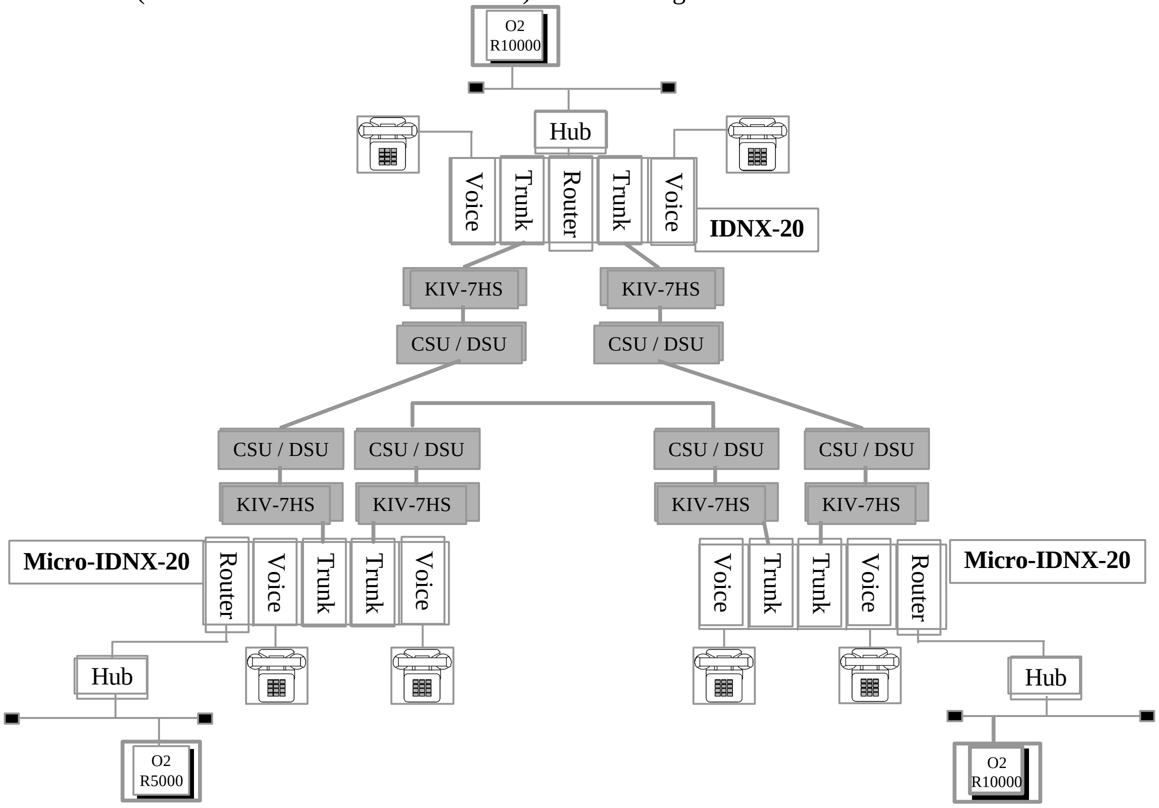


Figure 5. 3 node RTI test configuration with communications devices

In order to expand testing from the three owned SGI computers available at JADS to a total of six SGIs, JADS decided to lease three additional SGI O2 computers. These computers were delivered and installed in May 1998 to provide an expanded test bed of 6 computers to match the Phase 2 configuration.

Multi-Federate Software for Three-Node Tests

After characterizing the network and the RTI in the simple one-way tests, we wanted to determine whether the RTI would support the anticipated loads placed on it by the JADS federation. We wanted a test federate that could simulate these kinds of loads. The *testfed* federate was developed to satisfy these requirements. It can be executed on as many computers as necessary. The *testfed* federate accepts command line arguments that specify the characteristics of an instance of the federate. The user can specify the federate ID number (-f), the duration of the test (-d), the size of the attributes and interactions (-s), the rate that attributes are published (-r), the number of updates at the specified rate (-n), the amount of time the federate should wait before starting to publish at its specified rate (-w), and whether interactions should be published (-i). There is an additional argument (-c) that indicates which federate is the controller. There must be one and only one controller federate in the *testfed* federation. There is only one attribute and one interaction used by all federates. All federates subscribe to the attribute and the interaction.

Three-Node Three-Federate Tests with RTI 1.3-2EAV

Initially, we could only execute a three-federate test, because we only had three SGI O2 computers. For the three-federate test, we configured *testfed* on one computer to publish 11 attribute updates at 20 Hz (simulating the AFEWES federate). We configured another instance of *testfed* to publish 2 attribute updates at 20 Hz (simulating the federates at the JADS Albuquerque node). The third instance of *testfed* was configured to publish 1 attribute update at 20 Hz (simulating the ACETEF node). All three federates published interactions at approximately 1 Hz. The size of attributes and interactions was 121 bytes. Attributes were published best effort. Interactions were published reliable. We ran multiple tests with a duration of between two and five minutes. Initially, many attributes were lost at the very beginning of a test. We surmised that there may be a problem with all federates beginning to publish at their specified rate all at the same time so we implemented the wait option (-w). The wait option tells the federate how many seconds to send attribute updates at 1 Hz. When the wait period expires, the federate publishes attribute updates at its normal rate. After we began using the wait option, the missing attributes at the beginning of the test were eliminated.

Some runs have only a few attributes lost with maximum latency less than 45 ms. Other runs have up to 100 attributes lost with maximum interaction latency of over 1 second. We ran three tests with all federates on the same LAN. One of these tests had a maximum interaction latency of over 1.5 seconds. So, the problem does not seem to be caused by the communications equipment (routers, etc.).

Three-Node Six-Federate Tests with RTI 1.3-2EAV

After we leased three more SGI O2 computers, we ran a more realistic test with six federates on six computers on three network nodes. The six-federate tests produced a wide variety of results. We had a few runs where only one or two best effort attributes were lost and the maximum latency was less than 50 ms. There were some runs that had up to 100 attributes lost and an occasional high interaction latency of between 1 and 8 seconds and there were some runs that had federates that crashed. We reported these results to DMSO. Subsequently, the DMSO RTI development team found a software "bug" that limited the number of federates that could execute in a federation.

Three-Node Three-Federate Tests with RTI 1.3-2

We ran five tests with the same configuration: Federate 1 publishes 11 attribute updates at 20 Hz with interactions sent at 1 Hz; Federate 2 publishes 1 attribute update at 20 Hz with interactions sent at 1 Hz; and Federate 3 publishes 2 attribute updates at 20 Hz with interactions sent at 1 Hz. All five tests had at least one federate with a maximum latency greater than 70 ms. The largest maximum latency value was 1.79 seconds. There were two tests that had a maximum value over 250 ms.

Three-Node Six Federate Tests with RTI 1.3-2

We ran two 5-minute tests and six 3-minute tests with six federates on three nodes. There were no runs that had federates that crashed. Therefore, this problem appears to be fixed. However, in the one of the 5-minute tests, all of the federates had an interaction maximum latency over 3 seconds (the worst was 10 seconds). Five of the six federates in the second test had interaction maximum latencies above 700 ms (the worst was 2.2 seconds).

Lessons Learned

JADS experience with testing versions of the RTI proved to be an iterative process of identifying problems and implementing solutions. The problems related to the RTI were discussed directly with DMSO who worked with JADS to resolve the problems. This process reflects the type of RTI performance tuning JADS experienced.

Time-To-Live

In the initial tests we performed with RTI 1.0-2, best effort traffic was not received at any computer on a different LAN. Using a network sniffer to look at the network data packets, JADS discovered that the Time-To-Live value was set to 1. A packet's Time-To-Live value indicates how many hops it can take before it is discarded by the network. A value of 1 does not allow a packet to exit the LAN. A federation running with RTI 1.0-2 out of the box would not allow federates to communicate best effort traffic outside of a LAN. Using the JADS 2 node network configuration (shown in Figure 2) required network data packets to cross from one LAN through the routers (Micro-IDNX-20) to reach the test federate on another LAN mirroring the JADS EW phase 2 network architecture. JADS was provided a special library from DMSO that allowed us to use RTI 1.0-2 across our network communications gear. Subsequent versions of the RTI provide for a user-defined parameter value in the RID file to set Time-To-Live.

TCP No Delay and the Nagle Algorithm

The Nagle algorithm buffers small packets on the transmit side until an acknowledgment (ACK) is received from the previous transmit. On SGI computers, the network can wait up to 200 ms before sending the buffered packets. This explains the jump in latency at transmit rates over 5 Hz. By default, TCP sockets on SGIs run with the Nagle algorithm. To disable the Nagle algorithm, the programmer must specify TRUE for the socket option TCP_NODELAY.

Prior to RTI version 1.3-2, the RTI runs with a default setting for the TCP_NODELAY socket option. On the SGIs, the default value is FALSE. This means that the Nagle algorithm will be in effect for both attribute and interaction data sent reliable. If data is published using reliable transportation at data rates at or above 5 Hz, then the latency of the data is increased significantly. As a result of sharing this information with RTI developers, RTI version 1.3-2 sets the TCP_NODELAY option to TRUE, disabling the Nagle algorithm.

Tick

The tick function is how a federate transfers process control to the RTI so it can do its work. As we implemented and experimented with tick during initial test runs with each RTI release, we learned how important it is to understand how tick works in its various forms in order to tune a federation properly. The user's federates must constantly tick the RTI or nothing will happen in the federation. There are two variations to tick: one has no parameters (tick ( ), the other has a minimum and maximum value (tick (min, max). The tick function called with no parameters empties its queue before it returns to the federate which could starve the federate from getting its necessary processor time.

The tick function called with a minimum and maximum value (tick (min, max)) will stay in tick at least the amount of time specified by the minimum parameter, but no longer than the maximum parameter. If the RTI empties its queue before the minimum time elapses, it will try to sleep for the rest of the time. On an SGI, this is a problem because the minimum sleep (i.e., sginap or select) time is 10ms. As a result, if the RTI user specifies a minimum value of 10 ms and the RTI uses 9 ms to do its work, an SGI will sleep for an additional 10 ms. If zero or some small number is specified for the minimum, the RTI will not sleep. This can cause the federate/RTI to use as much as 90% of the CPU. We benefited greatly from open communication with RTI developers about features of tick and verifying the results we obtained from different settings. Unfortunately, we did not find any documentation source for tick features and tuning ideas. We advised developers that this information is very beneficial to all but the casual RTI user. Each federation and its architecture is different and it will require some experimentation by the federation developers to find the optimum use of tick.

Initial Publication Rates

When a federate starts, we found that it is best if it publishes some initial data at low data rates to set up the network. In the JADS tests with three federates (one published 11 updates at 20 Hz, one published 2 updates at 20 Hz, and one published one update at 20 Hz), best effort data was lost and reliable data had high latencies in the initial burst of data. When we added a 5 second delay at the start, during which the federates published data at 1 Hz, these startup problems were eliminated.

Fast *malloc*

There is a library on SGIs that provides a faster version of *malloc* (used to dynamically allocate memory). To use this library it must be linked with your software with the -*lmalloc* option. In an attempt to make it as efficient as possible, the JADS RTI logger was linked with this library. While running RTI tests linked with the logger, the federate would crash after it resigned from the federation. After speaking with the RTI development team, they said they were aware of problems using this library and recommended not using it.

Network Optimization

During a conversation to discuss the latency problems, representatives at ACETEF mentioned that their facility used Ethernet switches to eliminate problems that might be caused by collisions over a shared Ethernet configuration like the JADS test bed. JADS purchased and installed an 8-port Ethernet switch for use in the dedicated test bed. This increased the test bed LAN's performance well beyond that of the previous 10 Mbps, half duplex LAN. Recent two-federate testing with the *testfed* tool for a 5-minute test has shown that the Ethernet collision rate has dropped from about 500 - 1000 collisions to 0 collisions! While this LAN improvement has not eliminated all the latency problems observed during tests of RTI 1-3.2 (because Ethernet collisions were not the only cause of them), it seems to have significantly reduced, directly or indirectly, the frequency of the 1-second-class latency events.

All TCP Implementations Are Not the Same

The RTI is developed on Sun/Solaris machines. It has been optimized to run in this environment. The implementation of TCP on SGI/IRIX machines (and others) is different than the implementation of TCP on Sun/Solaris. Since TCP is a major Internet protocol, one might naively assume all TCP implementations were developed and tested to conform to the same "standards" and they all operate in more-or-less the same manner. JADS found information in the TCP literature[4] showing this naive view to be incorrect. For example, the current IRIX TCP on the SGI systems at JADS, AFEWES, and ACETEF is very likely to be one of the "Berkeley-derived implementations", but the current Solaris TCP used by the RTI's developer is very likely to be an "independent" implementation by Sun Microsystems. Furthermore, various TCP implementations are known to respond differently in time-critical situations. This suggests that distributed simulations running the same federates over the same LANs and WANs might exhibit different behavior on SGIs than on Sun/Solaris machines because of differences in the underlying TCP implementations.

Summary

This paper documents the methodology, accumulated experience, and current results of JADS RTI test activities conducted between March and July of 1998. In this time frame, the following RTI versions for SGI/IRIX 6.3 were tested:

RTI Version	**Date Released**
1.0-2	February 1998
1.3b	3 April 1998
1.3-2 EAV	15 May 1998
1.3-2	15 June 1998
1.3-4	September 1998

Based upon the latency values measured for the most recent RTI software release, further tests will be conducted as further resolution of latency problems are accomplished by DMSO and JADS. As documented in this paper, much has been accomplished and learned by both JADS and DMSO's RTI team from this effort. The progress made and lessons learned so far represent a significant advance in characterizing and reducing RTI latency as well as improving RTI features. However, the results measured to this point do not satisfy JADS criteria for success.

DMSO has provided significant support to address RTI problems as they were discovered. The DMSO release of the "formal" RTI version 1.3-2 for IRIX 6.3 SGI workstations is scheduled on 30 June 1998. JADS will conduct further tests of the RTI software with DMSO.

References

[1] D. Wright and C. Harris: "Testing RTI/Network Interactions for Latency", SISO Spring 98 Simulation Interoperability Workshop, paper 98S-SIW-152.
[2] D. Wright and C. Harris: "Determining and Expressing RTI Requirements", SISO Spring 98 Simulation Interoperability Workshop, paper 98S-SIW-158.
[3] W. Richard Stevens: "TCP Interactive Data Flow" TCP/IP Illustrated - Volume 1, Addison-Wesley, Massachusetts 1994.
[4] Vern Paxson: "Automated Packet Trace Analysis of TCP Implementations", Proceedings of SIGCOMM '97, technical paper.

8. Author Biographies

Clyde Harris is a Senior Systems Engineer for SAIC working with the JADS Joint Test Force in Albuquerque, NM, since 1994. He has over 27 years total experience in software and systems engineering involving communications systems and medical, command and control, intelligence, and logistics information systems for commercial business and DoD. His current area of focus spans over 8 years in test and evaluation of computer and communications systems and diverse applications.

Jerry Black is a Senior Software Engineer for SAIC in Albuquerque, New Mexico. He received a B.S. in Computer Science from the University of New Mexico in 1981. Mr. Black has over 17 years of experience developing software for DoD applications including flight software testing, imagery intelligence training, B-2 FOT&E, and real-time data collection. He has been supporting the JADS Joint Test Force as a software analyst responsible for time synchronization, data collection and analysis since 1995.